

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Craig Hansen et al.

Application No.: 10/757,939

Filed: January 16, 2004

For: Multithreaded Programmable Processor
and System with Partitioned Operations

Customer Number: 20277

Confirmation Number: 4645

Examiner: Jesse R. Moll

Technology Center/Art Unit: 2181

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF JOHN MOUSSOURIS UNDER 37 CFR § 1.131

Sir:

I, John Moussouris, hereby declare the following to be true:

BACKGROUND

1. I am a co-inventor of United States Patent number 10/757,939 (the '939 application). I understand that the '939 patent is currently being examined by the Patent Office. Among other things, this declaration describes certain activities that occurred prior to October, 1994 through August 16, 1995, the filing date of U.S. Patent Application Serial number 08/516,036. The '036 application matured into U.S. Patent number 5,742,840, which is a parent of the '939 patent. Attached to this declaration are various documents that evidence conception and diligence in reducing certain inventions claimed in the '939 patent to practice.
2. Prior to founding MicroUnity, I co-founded (and served as vice president of research and development at) MIPS Computer Systems, Inc. ("MIPS"), where I staffed

and managed the development of a RISC microprocessor and a math coprocessor, two chips that led the industry in performance for several years.

3. Before co-founding MIPS, I was an IBM Visiting Scholar at Stanford University's Center for Integrated Systems. At IBM T.J. Watson Research Lab in Yorktown, I was a founding member of a VLSI research project that developed a microprocessor based on the 801 RISC architecture. I also received an Outstanding Innovation Award for design of run-length-limited coding hardware used in IBM disk products. Before IBM, I was a research staff member at the MIT Laboratory for Computer Science.

4. I received my A.B. degree in Physics from Harvard College and a Masters degree in mathematics and a Doctoral degree in mathematical physics from Oxford University, which I attended as a Rhodes Scholar.

5. I founded MicroUnity in 1988. Prior to October, 1994, and through August 16, 1995, I was the CEO and Chairman of the Board of MicroUnity. As such, I have personal knowledge of MicroUnity's efforts toward implementing the Terpsichore system.

6. I am currently a shareholder in MicroUnity. I own approximately 69 percent of the shares of the company on a fully diluted basis. My current position at MicroUnity is Chairman of the Board.

7. All of the exhibits attached to this declaration are part of MicroUnity's records, and the exhibits are authentic and true copies of original documents (except for the redactions of dates in Exhibits A1 and A2, redactions of confidential/ privileged descriptions of the legal services provided to MicroUnity in Exhibit B, redactions of

employee names and social security numbers in Exhibit E2, and the addition of exhibit and page numbers used for ease of reference).

CONCEPTION

8. Along with my co-inventor, Craig Hansen, I conceived of a Multithreaded Programmable Processor and System with Partitioned Operations that included the following claims:

1. A programmable processor comprising:
 - a data path capable of transmitting data;
 - an external interface operable to receive data from an external source and communicate the received data over the data path;
 - a register file containing a plurality of registers each having a register width, the register file coupled to the data path and configured to support processing of a plurality of threads and to store a plurality of multiple-bit data elements in partitioned fields, each of the multiple-bit data elements having an elemental width smaller than the register width;
 - an execution unit coupled to the data path, the execution unit configured to execute a plurality of instruction streams from the plurality of threads, each instruction stream including a single instruction that specifies an arithmetic operation to cause multiple instances of the arithmetic operation to be performed, each instance of the arithmetic operation to be performed using a different one of the plurality of multiple-bit data elements in partitioned fields of at least one of the registers to produce a catenated result; and
 - wherein each of the multiple-bit data elements has an elemental width, and the data path has a data path width multiple times greater than the elemental width, to allow multiple-bit data elements used for the multiple instances of the arithmetic operation to be transmitted in parallel from the register file to the execution unit, and wherein the execution unit is operable to receive, in parallel, multiple-bit data elements for the multiple instances of the arithmetic operation and execute the multiple instances of the arithmetic instruction to produce the catenated result.
2. The processor of claim 1 wherein the execution unit comprises a pipeline having a plurality of stages and wherein the pipeline interleaves execution of instructions from the plurality of instruction streams.
3. The processor of claim 2 wherein the pipeline is operable to simultaneously contain states of execution of at least two instructions from different instruction streams.

4. The processor of claim 2 wherein execution of the instructions is interleaved in a round-robin manner.
5. The processor of claim 1 wherein the processor ensures only one thread from the plurality of threads can have an exception handled at any given time.
6. The processor of claim 1 further comprising a virtual memory addressing unit and a cache operable to store data communicated between the external interface and the data path.
7. The processor of claim 1 wherein the execution unit is further operable to, in response to decoding another single instruction specifying a first and a second register each containing a plurality of floating-point operands, multiply the plurality of floating-point operands in the first register by the plurality of floating-point operands in the second register to produce a plurality of products and provide the plurality of products to partitioned fields of a result register as a second catenated result.
8. A programmable processor comprising:
a data path capable of transmitting data;
an external interface operable to receive data from an external source and communicate the received data over the data path;
first and second register files containing a plurality of registers each having a register width, the first and second register files coupled to the data path and configured to support processing of first and second threads, respectively, and to store a plurality of multiple-bit data elements in partitioned fields, each of the multiple-bit data elements having an elemental width smaller than the register width;
an execution unit coupled to the data path, the execution unit configured to execute first and second instruction streams from the first and second threads, respectively, the first and second instruction streams each including a single instruction that specifies an arithmetic operation to cause multiple instances of the arithmetic operation to be performed, each instance of the arithmetic operation to be performed using a different one of multiple-bit data elements in partitioned fields of at least one of the registers to produce a catenated result; and
wherein each of the multiple-bit data elements has an elemental width, and the data path has a data path width multiple times greater than the elemental width, to allow multiple-bit data elements used for the multiple instances of the arithmetic operation to be transmitted in parallel from the first register file and from the second register file to the execution unit, and wherein the execution unit is operable to receive, in parallel, multiple-bit data elements for the multiple instances of the arithmetic operation and execute the multiple instances of the arithmetic instruction to produce the catenated result.

9. The processor of claim 8 wherein the execution unit comprises a pipeline having a plurality of stages and wherein the pipeline interleaves execution of instructions from the first instruction stream with instructions from the second instruction stream.

10. The processor of claim 9 wherein the pipeline is operable to simultaneously contain states of execution of an instruction from the first instruction stream and an instruction from the second instruction stream.

11. The processor of claim 9 wherein execution of the instructions is interleaved in a round-robin manner.

12. The processor of claim 9 wherein the execution unit is further operable to, in response to decoding another single instruction specifying a first and a second register each containing a plurality of floating-point operands, multiply the plurality of floating-point operands in the first register by the plurality of floating-point operands in the second register to produce a plurality of products and provide the plurality of products to partitioned fields of a result register as a second catenated result.

13. A data processing system comprising:

- (a) a bus coupling components in the data processing system;

- (b) an external memory coupled to the bus;

- (c) a programmable microprocessor coupled to the bus and capable of operation independent of another host processor, the microprocessor comprising:

- a data path capable of transmitting data;

- an external interface operable to receive data from an external source and communicate the received data over the data path;

- a register file containing a plurality of registers each having a register width, the register file coupled to the data path and configured to support processing of a plurality of threads and to store a plurality of multiple-bit data elements in partitioned fields, each of the multiple-bit data elements having an elemental width smaller than the register width;

- an execution unit coupled to the data path, the execution unit configured to execute a plurality of instruction streams from the plurality of threads, each instruction stream including a single instruction that specifies an arithmetic operation to cause multiple instances of the arithmetic operation to be performed, each instance of the arithmetic operation to be performed using a different one of the plurality of data elements in partitioned fields of at least one of the registers to produce a catenated result; and

- wherein each of the multiple-bit data elements has an elemental width, and the data path has a data path width multiple times greater than the elemental width, to allow multiple-bit data elements used for the multiple instances of the arithmetic operation to be transmitted in parallel from the register file to the execution unit, and wherein the execution unit is

operable to receive, in parallel, multiple-bit data elements for the multiple instances of the arithmetic operation and execute the multiple instances of the arithmetic instruction to produce the catenated result.

14. The system of claim 13 wherein the execution unit comprises a pipeline having a plurality of stages and wherein the pipeline interleaves execution of instructions from the plurality of instruction streams.

15. The system of claim 14 wherein the pipeline is operable to simultaneously contain states of execution of at least two instructions from different instruction streams.

16. The system of claim 14 wherein execution of the instructions is interleaved in a round-robin manner.

17. The system of claim 13 wherein the processor ensures only one thread from the plurality of threads can have an exception handled at any given time.

18. The system of claim 13 further comprising a virtual memory addressing unit and a cache operable to store data communicated between the external interface and the data path.

19. The system of claim 13 wherein the execution unit is further operable to, in response to decoding another single instruction specifying a first and a second register each containing a plurality of floating-point operands, multiply the plurality of floating-point operands in the first register by the plurality of floating-point operands in the second register to produce a plurality of products and provide the plurality of products to partitioned fields of a result register as a second catenated result.

20. A data processing system comprising:

(a) a bus coupling components in the data processing system;

(b) an external memory coupled to the bus;

(c) a programmable microprocessor coupled to the bus and capable of operation independent of another host processor, the microprocessor comprising:

a data path capable of transmitting data

an external interface operable to receive data from an external source and communicate the received data over the data path;

first and second register files containing a plurality of registers each having a register width, the first and second register files coupled to the data path and configured to support processing of first and second threads, respectively, and to store a plurality of multiple-bit data elements in partitioned fields, each of the multiple-bit data elements having an elemental width smaller than the register width;

an execution unit coupled to the data path, the execution unit configured to execute first and second instruction streams from the first and second threads, respectively, the first and second instruction streams each including a single instruction that specifies an arithmetic operation to cause multiple instances of the arithmetic operation to be performed, each instance of the arithmetic operation to be performed using a different one of the plurality of multiple-bit data elements in partitioned fields of at least one of the registers to produce a catenated result; and

wherein each of the multiple-bit data elements has an elemental width, and the data path has a data path width multiple times greater than the elemental width, to allow multiple-bit data elements used for the multiple instances of the arithmetic operation to be transmitted in parallel from the first register file and from the second register file to the execution unit, and wherein the execution unit is operable to receive, in parallel, multiple-bit data elements for the multiple instances of the arithmetic operation and execute the multiple instances of the arithmetic instruction to produce the catenated result.

21. The system of claim 20 wherein the execution unit comprises a pipeline having a plurality of stages and wherein the pipeline interleaves execution of instructions from the first instruction stream with instructions from the second instruction stream.

22. The system of claim 21 wherein the pipeline is operable to simultaneously contain states of execution of an instruction from the first instruction stream and an instruction from the second instruction stream.

23. The system of claim 21 wherein execution of the instructions is interleaved in a round-robin manner.

24. The system of claim 21 wherein the execution unit is further operable to, in response to decoding another single instruction specifying a first and a second register each containing a plurality of floating-point operands, multiply the plurality of floating-point operands in the first register by the plurality of floating-point operands in the second register to produce a plurality of products and provide the plurality of products to partitioned fields of a result register as a second catenated result.

25. The processor of claim 1 wherein the arithmetic operation comprises an integer operation.

26. The processor of claim 1 wherein the arithmetic operation comprises a floating-point operation.

27. A programmable processor comprising:
a data path capable of transmitting data;

an external interface operable to received data from an external source and communicate the received data over the data path;

a register file containing a plurality of registers each having a register width, the register file coupled to the data path and configured to support processing of a plurality of threads and to store a plurality of multiple-bit data elements in partitioned fields, each of the multiple-bit data elements having an elemental width smaller than the register width;

an execution unit coupled to the data path, the execution unit configured to execute a plurality of instruction streams from the plurality of threads, each instruction stream including a single instruction that specifies a floating-point arithmetic operation to cause multiple instances of the floating-point arithmetic operation to be performed, each instance of the floating point arithmetic operation to be performed using a different one of the plurality of multiple-bit data elements in partitioned fields of at least one of the registers to produce a catenated result; and

wherein each of the multiple-bit data elements has an elemental width, and the data path has a data path width multiple times greater than the elemental width, to allow multiple-bit data elements used for the multiple instances of the floating-point arithmetic operation to be transmitted in parallel from the register file to the execution unit, and wherein the execution unit is operable to receive, in parallel, multiple-bit data elements for the multiple instances of the floating-point arithmetic operation and execute the multiple instances of the floating-point arithmetic instruction to produce the catenated result.

28. A data processing system comprising:

(a) a bus coupling components in the data processing system;

(b) an external memory coupled to the bus;

(c) a programmable microprocessor coupled to the bus and capable of operation independent of another host processor, the microprocessor comprising:

a data path capable of transmitting data;

an external interface operable to receive data from an external source and communicate the received data over the data path;

a register file containing a plurality of registers each having a register width, the register file coupled to the data path and configured to support processing of a plurality of threads and to store a plurality of multiple-bit data elements in partitioned fields, each of the multiple-bit data elements having an elemental width smaller than the register width;

an execution unit coupled to the data path, the execution unit configured to execute a plurality of instruction streams from the plurality of threads, each instruction stream including a single instruction that specifies a floating-point arithmetic operation to cause multiple instances of the floating-point arithmetic operation to be performed, each instance of the floating-point arithmetic operation to be performed using a different one of the plurality of multiple-bit data elements in partitioned fields of at least one of the registers to produce a catenated result; and

wherein each of the multiple-bit data elements has an elemental width, and the data path has a data path width multiple times greater than the elemental width, to allow multiple-bit data elements used for the multiple instances of the floating-point arithmetic operation to be transmitted in parallel from the register file to the execution unit, and wherein the execution unit is operable to receive, in parallel, multiple-bit data elements for the multiple instances of the floating-point arithmetic operation and execute the multiple instances of the floating-point arithmetic instruction to produce the catenated result.

9. It is my understanding that the claims 2-5, 9-12, 14-17 and 21-23 recited above have been rejected in the outstanding office action based on Cray, Jr. (U.S. Patent Number 4,128,880, hereinafter “Cray”) in view of Chen *et al.* (U.S. Patent Number 4,771,900, hereinafter “Chen”) in further view of Laudon *et al.* (U.S. Patent Number 5,938,756, hereinafter “Laudon”).

10. The conception (before the publication date of Laudon, *i.e.*, before October or November, 1994) of the above-recited claims of the Multithreaded Programmable Processor and System with Partitioned Operations is shown in Exhibits A1 and A2, attached hereto. The dates of Exhibit A1 and A2 are prior to October, 1994. However, the dates have been redacted in accordance with standard practice and indicated with the phrase “REDACTED”.

Exhibits A1 and A2 – Presentation to Cray Research, Inc. and the pre-filing version of the Terpsichore System Architecture Manual

11. Exhibit A1 is a presentation made to Cray Research, Inc. pursuant to a confidentiality agreement between MicroUnity and Cray Research, Inc., with dates redacted. The presentation is a technology overview of the Terpsichore System Architecture and further describes certain subject matter disclosed and claimed in the ‘939 application.

12. Exhibit A2 is a version of the Terpsichore System Architecture manual that was authored by Mr. Hansen under my supervision prior to October, 1994, with the date redacted. The manual describes certain subject matter disclosed and claimed in the '939 application. This manual was updated while work on the Terpsichore system progressed. The August 2, 1995 version of this manual was filed as an appendix to the '036 application.

13. Exhibits A1 and A2 are described in detail in Mr. Hansen's declaration filed concurrently herewith. In particular, I refer the PTO to paragraph 11 of Mr. Hansen's declaration.

14. As indicated above, Exhibits A1 and A2 were prepared before October, 1994, prior to which I spent a considerable amount of time conceiving and developing the Terpsichore System in conjunction with Mr. Hansen. Based on the accompanying Exhibits, Mr. Hansen and I conceived the fundamental features of the Terpsichore System prior to October, 1994, which we believed would work for their intended purpose once sufficient prototyping and testing efforts were performed.

DILIGENCE

15. Just prior to October, 1994, and through August 16, 1995, I and others at MicroUnity, as well as MicroUnity's legal patent prosecution team, were diligent in our efforts to perform detailed design, to build and to test the Terpsichore system and to file the '036 application, which eventually matured into the '840 patent, which is a parent of the '939 patent. MicroUnity's efforts to further design, build and test the Terpsichore system included the incorporation of features conceived prior to October, 1994 into integrated circuitry implementing the Terpsichore system and the development work

associated with such incorporation.

16. The evidence showing diligence includes: attorney billing records from MicroUnity's patent prosecution team [Exhibit B]; email communications among the MicroUnity design team making modifications to the electronic databases [Exhibit C]; modifications made to the electronic databases used to manufacture integrated circuits implementing the Tersichore system [Exhibit D]; and MU payroll records showing salary payments to team members [Exhibit E].

Exhibit B – Attorney Billing Records

17. Exhibit B is a collection of invoices from the law firm of Willian, Brinks, Hofer, Gilson & Lione, MicroUnity's patent prosecution counsel. These documents are explained in detail in paragraph 15 of Mr. Hansen's declaration filed concurrently herewith, and I refer the PTO to that paragraph of Mr. Hansen's declaration.

Exhibits C6-C17 – Emails among the MicroUnity Design Team

18. Further evidence of MicroUnity's efforts to implement the Tersichore system in integrated circuit form is shown in email communications among the members of MicroUnity's design team from the time just prior to October, 1994, through August 16, 1995. These emails are grouped by month and are attached hereto as Exhibits C6-C17. These emails are explained in detail in paragraphs 16-29 of Mr. Hansen's declaration filed concurrently herewith, and I refer the PTO to those paragraphs of Mr. Hansen's declaration.

Exhibits D1 and D23-D70 – Logs of Modifications to the Electronic Databases

19. MicroUnity's Terpsichore system was intended to be implemented in integrated circuit form. From a time prior to October, 1994, through August 16, 1995, the individuals on the MicroUnity design team spent a substantial amount of their time building elaborate databases (sometimes called "tapeouts" or "physical layouts") for the Terpsichore system. Exhibits D23-D70 represent weekly logs of modifications to the electronic databases for the Terpsichore system from September 20, 1994, through August 16, 1995. Exhibit D1 is a summary, on a weekly basis, of the number of modifications made to these electronic databases during this time period. These documents are explained in detail in paragraphs 30-83 of Mr. Hansen's declaration filed concurrently herewith, and I refer the PTO to those paragraphs of Mr. Hansen's declaration.

Exhibit E – Payroll Records

20. Exhibit E reflects various MicroUnity payroll records from April 1, 1994 through August 15, 1995. These documents show that MicroUnity consistently spent between approximately \$250,000 monthly on payroll for the design team, with a total expenditure of approximately \$3M for the time period from October 1, 1994 until August 16, 1995 to develop the Terpsichore system. These documents are explained in detail in paragraph 84 of Mr. Hansen's declaration filed concurrently herewith, and I refer the PTO to that paragraph of Mr. Hansen's declaration.

21. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and the these statements are made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of

the United States Code, and that such willful false statements may jeopardize the validity of the application, and any patent issuing thereon, or any patent to which this declaration is directed.

September 17, 2009

Date

A handwritten signature in black ink, appearing to read "John Moussouris", is written over a horizontal line.

John Moussouris